Rechnerstrukturen

Vorlesung im Sommersemester 2006

Prof. Dr. Wolfgang Karl

Universität Karlsruhe (TH)

Fakultät für Informatik

Institut für Technische Informatik

Kapitel 1: Grundlagen

1.3 Bewertung der Leistungsfähigkeit

- Messung während des Betriebs von Anlagen
  - Monitore
    - Aufzeichnungselemente, die zum Zweck der Rechnerbewertung die Verkehrsverhältnisse während des normalen Betriebs beobachten und untersuchen.
    - Hardware-Monitore
      - Unabhängige physikalische Geräte
      - Keine Beeinflussung
    - Software-Monitore
      - Einbau in das Betriebssystem
      - Beeinträchtigung der normalen Betriebsverhältnisse

- Monitore
  - Aufzeichnungstechniken:
    - Kontinuierlich oder sporadisch
    - Gesamtdatenaufzeichnung (Tracing)
    - Realzeitauswertung
    - Unabhängiger Auswertungslauf (Post Processing)

Bewertung der Leistungsfähigkeit

- Modelltheoretische Verfahren
  - Unabhängig von der Existenz eines Rechners
  - – Modellbildung
    - aufgrund von Annahmen über die Struktur und Betrieb eines Rechners und über die Prozesse
    - Darstellung der für die Analyse relevanten Merkmale des Systems:
      - – Systemkomponenten
      - – Datenverkehr zwischen den Systemkomponenten
    - Abstrahierung komplexer Systeme
      - – Nur die interessierenden Größen werden erfasst
    - Ziel:
      - – Aufdecken von Beziehungen zwischen Systemparametern
      - – Ermitteln von Leistungsgrößen (Auslastung von Prozessoren und Kanälen, mittlere Antwortzeiten, Warteschlangenlängen, ...)

- Modellbildung
  - Analytische Methoden
    - versuchen auf mathematischem Weg, Beziehungen zwischen relevanten Leistungskenngrößen und fundamentalen Systemparametern herzuleiten
    - oft nur minimaler Aufwand, aber dafür weniger aussagekräftig
  - Warteschlangenmodelle
    - Leistungsanalyse von Rechensystemen
  - Petrinetze
    - theoretische Untersuchungen
  - Diagnosegraphen
    - Zuverlässigkeitsanalysen
  - Netzwerkflussmodelle
    - Kapazitätsüberlegungen

- Modellbildung
  - Analytische Methoden
    - Beispiel Warteschlangenmodelle
      - Deterministische Warteschlangenmodelle
        - » Beispiele für Systemparameter: Rechenzeit, Gerätebedienzeit, Ankunftszeit eines Jobs
        - » Feste Werte → deterministische Ergebnisse für die Leistungsgrößen
      - Stochastische Warteschlangenmodelle
        - » Systemparameter statistisch verteilt, mit vorgegebenen Mittelwerten, Verteilungsfunktionen → statistisch verteilte Leistungsgrößen

- Modellbildung
  - Analytische Methoden
    - Beispiel Warteschlangenmodelle
      - Operationelle Warteschlangenmodelle
        - » Systemparameter: Gemessene Werte aus der Beobachtung eines Systems in einem festen Zeitintervall
        - » Vereinfachung der Gleichungen für die Bestimmung von Leistungsgrößen
        - » Relativ gute Aussagen über das Leistungsverhalten des Systems bei geeignet gewähltem Zeitintervall

- Modellbildung
  - Simulation
    - Vorgänge in einem Rechensystem werden nachgebildet
    - Verwendung üblicher Programmiersprachen oder spezieller Simulationssprachen
    - Verhalten des Simulationsmodells in Bezug auf die relevanten Parameter entspricht weitgehend dem Verhalten des realen Systems
    - Ermittlung der für die Leistungsbewertung interessierenden Größen

- Modellbildung
  - Simulation
    - Deterministische Simulation
      - Alle an dem Modell beteiligten Größen sind exakt definiert oder berechenbar
    - Stochastische Simulation
      - Verwendung von zufallsabhängigen Größen
      - Oft: Einsatz von Zufallsgeneratoren
    - Aufzeichnungsgesteuerte Simulation
      - Verwendung von gemessenen Werten
      - Erfasst an aufgrund der Beobachtung eines Systems in einem festen Zeitintervall

- Modellbildung
  - Simulation
    - Nachteile:
      - Vorbereitung und Ausführung der Simulationsmodelle zeitaufwendig und teuer
      - Planung der Experimente muss sorgfältig durchgeführt werden
      - Auswertung und Interpretation der Ergebnisse nicht immer einfach
    - Beispiel:
      - SimpleScalar Tool Set (http://www.simplescalar.com)
        - » „Standard“-Werkzeug zur Simulation von superskalarer Mikroprozessoren

- Modellbildung
  - Simulation vs. Analytische Methoden
    - Simulation:
      - Realistischere Annahmen über das System bei der Simulation
      - Berücksichtigung vieler verschiedener Systemgrößen
      - Abdeckungen verschiedener Anwendungsbereiche
    - Können sich gegenseitig gut ergänzen

- Zusammenfassung
  - Rechnerleistung abhängig von
    - Blickwinkel (Nutzer, Administrator)
    - Nutzung des Rechners (Programm)
    - System- und Betriebssoftware
    - Compilertechnologie
    - Befehlssatz
    - Mikroarchitektur
    - Halbleitertechnologie

- Zusammenfassung
  - Einfache Maßzahlen sind wenig aussagekräftig
    - Beschreiben nur einzelne Aspekte der Leitung
    - Nie isoliert betrachten
  - Zur Rechnerauswahl / Bewertung von Systemen vor bzw. beim Kauf:
    - Benchmarks
      - Problem: Aufwand vs. Repräsentativität
      - De-facto-Standard für CPU-Leistung: CPU2000 Benchmark-Suite

- Zusammenfassung
  - Bewertung von Systemen im Betrieb
    - Monitore: Messung und Auswertung des Verhaltens
  - Bewertung von Systemen, die noch nicht existieren
    - Analytische Methoden und Simulation
    - Problem: Aufwand vs. Genauigkeit

- Übersicht über Rechnerbewertungsverfahren

	Auswertung von Hardware-maßen und Parametern	**Laufzeit-messungen bestehender Programme**	**Messungen während des Betriebs der Anlagen**	**Modell-theoretische Verfahren**
Rechnerauswahl	Maßzahlen für die Operations-geschwindingkeit Mixe Kernprogramme	Benchmarks		
Rechner-„Tuning“			Hardware-Monitore Software-Monitore	
Rechnerentwurf				Analytische Methoden Simulation

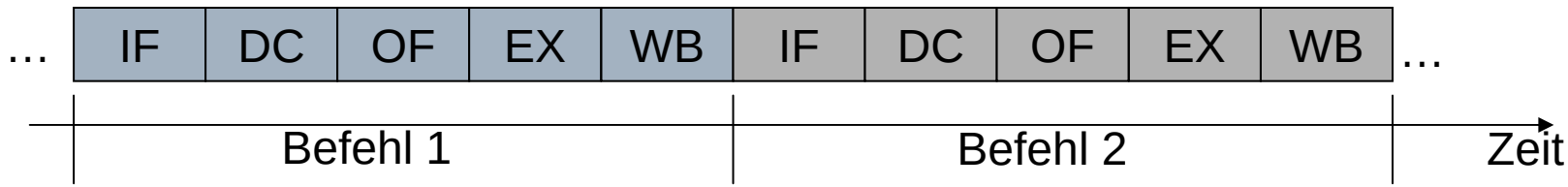
Kapitel 1: Grundlagen

1.4 Parallelverarbeitung

- Sequentielle Sichtweise
  - Berechnungen werden schrittweise oder seriell ausgeführt
  - Algorithmen sind als Folge von Berechnungsschritten organisiert
  - Sequentielle Programme
  - Schrittweise Ausführung einer Folge von Befehlen auf einer sequentiellen Maschine
- Beschleunigung der Ausführung
  - Erhöhung der Taktfrequenz
  - Parallele Ausführung der Aufgaben

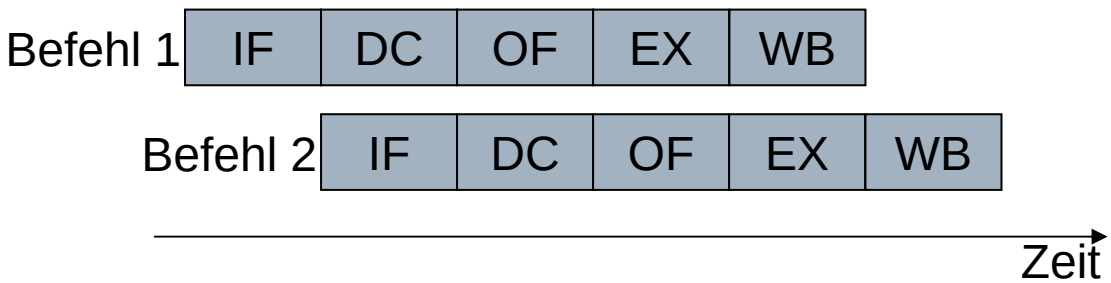
- Parallelverarbeitung
  - Algorithmenentwurf
    - Muss viele unabhängige Operationen umfassen
  - Struktur der Programmiersprache
    - Muss die Spezifikation oder die automatische Identifizierung paralleler Operationen ermöglichen
  - Rechnerarchitektur
    - Gleichzeitige Ausführung paralleler Operationen

- Sequentieller Rechner
  - Sequentielle Ausführung einer Folge von Maschinenbefehlen
    - Maschinenbefehlszyklus

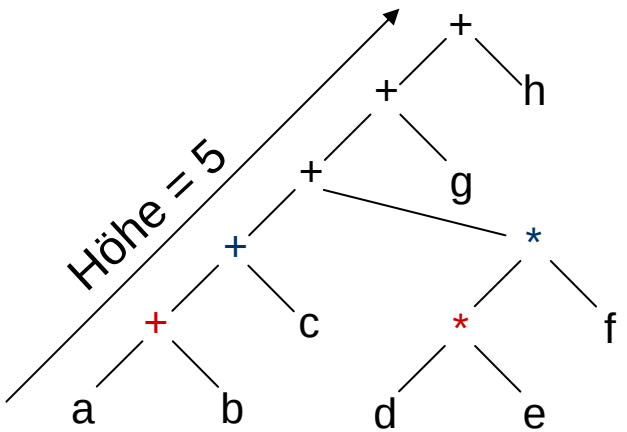


IF: Befehl holen
DK: Befehl dekodieren
OF: Operanden bereitstellen
EX: Befehl ausführen
WB: Ergebnis speichern

- Pipelining des Maschinenbefehlszyklus
  - Überlappte Ausführung der Phasen des Maschinenbefehlszyklus



- Gleichzeitige Ausführung von Operationen
  - Beispiel: Auswertung des arithmetischen Ausdrucks *EXP= a+b+c+(d*e*f)+g+h* durch den Compiler von links nach rechts



Seq. Ausführung
S1: t1 = a + c
S2: t1 = t1 + c
S3: t2 = d * e
S4: t2 = t2 * f
S5: t1 = t1 + t2
S5: t1 = t1 + g
S6: t1 = t1 + h

Parallele Ausf.
S1: t1 = a + c; S3: t2 = d * e
S2: t1 = t1 + c; S4: t2 = t2 * f
S5: t1 = t1 + t2
S5: t1 = t1 + g
S6: t1 = t1 + h

  - Höhe:
    - längster Pfad von der Wurzel zum Blatt
    - Minimale Anzahl von Schritten zur Auswertung des Ausdrucks

- Formen des Parallelismus
  - Nebenläufigkeit
    - Eine Maschine arbeitet nebenläufig, wenn die Objekte vollständig gleichzeitig abgearbeitet werden.
  - Pipelining
    - Pipelining auf einer Maschine liegt dann vor, wenn die Bearbeitung eines Objektes in Teilschritte zerlegt und diese in einer sequentiellen Folge (Phasen der Pipeline) ausgeführt werden. Die Phasen der Pipeline können für verschiedene Objekte überlappt abgearbeitet werden. (Bode 95)

- Ebenen der Parallelität
  - Programmebene
    - Parallele Verarbeitung verschiedener Programme
    - Vollständig unabhängige Einheiten
      - ohne gemeinsame Daten
      - wenig oder keine Kommunikation und Synchronisation
    - Parallelverarbeitung wird vom Betriebssystem organisiert

- Ebenen der Parallelität
  - Prozessebene (Task-Ebene)
    - Programm wird in Anzahl parallel ausführbarer Prozesse zerlegt
      - Prozess: schwergewichtiger Prozess (heavy-weighted process), Beispiel: UNIX-Prozesse
        - » Besteht aus vielen sequentiell ausgeführten Befehlen und umfasst eigene Datenbereiche
      - Synchronisation und Kommunikationsaufwand
    - Betriebssysteme unterstützt Parallelverarbeitung durch Primitive zur Prozessverwaltung, Prozess-Synchronisation, Prozesskommunikation

- Ebenen der Parallelität
  - Blockebene
    - leichtgewichtige Prozesse (Threads)
      - Bestehen jeweils aus sequentiell ausgeführten Befehlen teilen sich gemeinsamen Adressraum
      - Beispiel: Threads gemäß POSIX 1003.a Standard in mehrfädigen (multithreaded) Betriebssystemen
      - Synchronisation über Schlossvariablen (mutex), und Bedingungsvariablen (condition variables) oder darauf aufbauenden Synchronisationsmechanismen
      - Kommunikation über gemeinsame Daten
      - Aufwand für Thread-Erzeugung und-Beendigung, Thread-Wechsel geringer

- Ebenen der Parallelität
  - Blockebene
    - Anweisungsblöcke
      - Innere und äußere parallele Schleifen in FORTRAN-Dialekten
      - Verwendung von Microtasking
      - Hohes Parallelitätspotential durch parallel ausführbare Schleifeninterationen

- Ebenen der Parallelität
  - Anweisungs- oder Befehlsebene
    - Parallele Ausführung einzelner Maschinenbefehle oder elementarer Anweisungen
    - Optimierende (parallelisierende) Compiler für VLIW-Prozessoren oder Anwendung der Superskalartechnik in superskalaren Mikroprozessoren
      - Analyse der sequentiellen Befehlsfolge
      - Umordnen und Parallelisieren der Befehle
    - Datenflusssprachen und funktionale Programmiersprachen erlauben explizite Spezifikation der Parallelität

- Ebenen der Parallelität
  - Suboperationsebene
    - Elementare Anweisung wird durch Compiler oder durch die Maschine in Suboperationen aufgebrochen, die parallel ausgeführt werden
      - Vektoroperationen
        - » Überlappte Ausführung auf Vektorrechner in Vektorpipeline
        - » Komplexe Datenstrukturen (Matrizen, Vektoren, Datenströme) sind in höherer Programmiersprache verfügbar oder werden von einem parallelisierenden, vektorisierenden Compiler aus einer sequentiellen Programmiersprache generiert
        - » Beispiel: Vektoraddition C = A + B statt Abarbeitung einer Schleife

- Körnigkeit der Parallelität
  - Die Körnigkeit oder Granularität (grain size) ergibt sich aus dem Verhältnis von Rechenaufwand zu Kommunikations- oder Synchronisationsaufwand. Sie bemisst sich nach der Anzahl der Befehle in einer sequentiellen Befehlsfolge.
  - Programm-, Prozess- und Blockebene werden häufig auch als grobkörnige (large grained) Parallelität,
  - die Anweisungsebene als feinkörnige (finely grained) Parallelität bezeichnet.
  - Seltener wird auch von mittelkörniger (medium grained) Parallelität gesprochen, dann ist meist die Blockebene gemeint.

- Techniken der Parallelarbeit vs. Parallelitätsebenen

Parallelarbeitstechniken	Programmebene	Prozeßebene	Blockebene	Anweisungsebene	Suboperationsebene
Techniken der Parallelarbeit durch Rechnerkopplung					
Hyper- und Metacomputer	X	X			
Workstation-Cluster	X	X			
Techniken der Parallelarbeit durch Prozessorkopplung					
Nachrichtenkopplung	X	X			
Speicherkopplung (SMP)	X	X	X		
Speicherkopplung (DSM)	X	X	X		
Grobkörniges Datenflußprinzip		X	X		
Techniken der Parallelarbeit in der Prozessorarchitektur					
Befehlspipelining				X	
Superskalar				X	
VLIW				X	
Überlappung von E/A- mit CPU-Operationen				X	
Feinkörniges Datenflußprinzip				X	
SIMD-Techniken					
Vektorrechnerprinzip					X
Feldrechnerprinzip					X

Quelle: Ungerer, T. : Skript Rechnerstrukturen, SS 2000

Einführung: Literatur

- Theo Ungerer: Parallelrechner und parallele Programmierung, Kap.1.1 – 1.3